

# Bringing Particle Accelerator MC Integrations to HPC

Carlo Graziani

4 April 2018

## 1 Introduction

This memo describes my understanding of the situation respecting the current state-of-the-art in computation of reaction rates at HEP experiments, and of some possible approaches to scaling those computations to HPC platforms. The basis for this understanding is the March 13 conversation with Stefan Hoeche, which lifted part (though not all) of the fog surrounding the rather complicated machinery used to compute those reaction rates now.

My understanding of what the computation is supposed to do is as follows:

The reaction rates represent the process of two particles interacting through standard model interactions, yielding  $N$  particles after the reaction is complete. The typical  $N$  are 2-5. Each value of  $N$ , together with the identification of 2 in and  $N$  out particle types, corresponds to a single process requiring a separate computation.

The computation is a phase space integration of a transition probability. The phase space represents the possible momenta and energies of the outbound particles (the inbound particles are considered to have known total energy, and are in their center-of-momentum frame). Each outbound particle has 3 kinetic degrees of freedom (its momentum components), and the overall reaction must conserve total momentum and energy (4 conditions). The dimension of the phase space is therefore  $3N - 4$ .

The integrand is the probability  $P$  of a transition from the inbound particles to the outbound particles in a differential element of phase space. This probability is a positive-definite density, obtained by taking the square magnitude of a quantum transition amplitude – the S-matrix element. That is,  $P = |S|^2$ , where  $S$  is a complex-valued function of phase space.

The S-matrix element is computed in perturbation theory by adding up  $M$  complex-valued terms called transition amplitudes, where  $M \sim 10^4$ . Each transition amplitude is represented by a certain Feynman diagram (a pictorial representation of a possible reaction that encodes instructions on how to compute the corresponding amplitude). The number of amplitudes is

large because of the high order in perturbation theory that is required to obtain the desired accuracy – the higher the perturbation-theoretic order, the more terms result.

Each amplitude has an individual phase-space structure. Some of them are fairly regular, others have integrable singularities (after pathological singularities have been removed). Each one is relatively simple, and would not present integration problems alone. The complicated phase-space structure of the transition probability integrand is the result of summing thousands of them to compute  $P = |S|^2$ .

In fact, the very complicated machinery (importance sampling + Vegas Algorithm) that has been developed to perform the integrations is driven by this very complicated behavior of  $|S|^2$ . In the case of  $N = 5$ , one must integrate over an 11-dimensional space filled with very disordered local structure, including integrable singularities and experimental “no-go” areas (which correspond to regions where the outbound particles are undetectable due to experimental constraints).

In addition to which, the cost of computing the integrand is highly inhomogeneous, since there are regimes in which terms in certain amplitudes nearly cancel, leading to catastrophic loss of precision. These regions, when noticed by the libraries that compute the amplitudes, trigger a switch in numerical precision, from 8-byte to 16-byte floating-point, which leads to a dramatic slowdown in evaluation time for the integrand query. This phenomenon is termed “instability” by the programmers who deal with it, although I believe this is probably improper terminology.

One result of this effect is that the transition of this computation to HPC resources has been difficult to implement, because it is difficult to efficiently distribute an unpredictable computational cost across HPC nodes.

This summarizes the current situation as I understand it. What follows are questions and speculations about possible ways forward.

## 2 Can The Integrand Computation Be Re-Factored?

The current approach to computing the integrand  $|S|^2$  is to compute the  $M$  complex amplitudes  $A_k$ ,  $k = 1, \dots, M$ , and sum them into a single complex number  $S = \sum_k A_k$ , which is then squared. The alternative approach would be to expand the sum,

$$\begin{aligned} |S|^2 &= \left| \sum_{k=1}^M A_k \right|^2 \\ &= \sum_{k=1}^M |A_k|^2 + 2 \sum_{k=2}^M \sum_{l=1}^{k-1} \Re(A_k^* A_l). \end{aligned} \quad (2.1)$$

The current approach has an obvious attraction over the alternative represented by Equation (2.1), in that after the  $A_k$  are computed it requires only about  $M$  arithmetic operations to assemble  $|S|^2$ . The alternative requires about  $\frac{3}{2}M^2$  arithmetic operations. For  $M \sim 10^4$ , one

might be regarded as slightly crazy (or as having more computing power than sense, anyway) for selecting the second alternative.

However, note that the individual terms in the sums of Equation (2.1) *have much simpler phase space structure* than the does the full transition probability  $|S|^2$ , because, after all, each amplitude  $A_k$  is a relatively simple function of phase space. So there is potentially value to breaking out the sum in this way, not to sum them directly, but rather to integrate them individually and sum the result. Each integration would presumably be much cheaper than the one-time integration of the full transition probability, because importance sampling would become much easier.

A refinement of this approach could be based on what I believe Stefan told us (if I understood him correctly) about the structure of the amplitudes  $A_k$ . Apparently, many of them share very similar phase-space structure, and could in principle be grouped together by structure. If  $I_\mu \equiv \{k_1^{(\mu)}, k_2^{(\mu)}, \dots, k_{M_\mu}^{(\mu)}\}$   $\mu = 1, \dots, P$  are sets of indices such that all the amplitudes  $A_{k_v^{(\mu)}}$  belong to structure class  $\mu$ , one could form the complex sums  $B_\mu = \sum_{v=1}^{M_\mu} A_{k_v^{(\mu)}}$ , and have  $S = \sum_{\mu=1}^P B_\mu$ . If  $P \ll M$  (and I seem to remember Stefan telling us that  $M/P \sim 10$ ), then the broken-out computation would only require  $O(P^2)$  integrations instead of  $O(M^2)$  integrations, which might constitute a considerable saving.

It is also worth noting that while  $M$  is fixed by the accuracy requirement, the number of cores available for computation scales according to the resources allocated to the task. So even if it turned out that the “break out the amplitudes” strategy is a loser on the computational resources currently used for the computation, as the number of cores is increased there necessarily comes a point where the new strategy starts to dominate the old one, at least so long as the computations may be distributed in a balanced manner.

### 3 Instability?

I was a bit perplexed at the part of the discussion with Stefan where the “instability” of the integrand computation came up. As I understand matters, this effect is really due to sudden loss of precision due to cancellations in quantum propagators (possibly due to resonances?). The “instability” (in computational cost of the integrand) arises in consequence of the computational strategy adopted to address the situation, which is to shift the calculation from 8-byte to 16-byte precision on the fly. This shift bogs down the rate at which the result can be computed.

The reason this is perplexing is that in my own experience, I have never found it necessary to change computational precision for the sake of addressing a near-cancellation of terms. Rather, I have always found it possible to rewrite the offending expression so that it can be correctly computed in the “unstable” regime.

As an example (which is probably oversimplified but which gives an idea of what I mean) consider the expression  $x - \sin x$ . This is may be computed to reasonable precision with 8-byte floating point arithmetic for  $|x| \gtrsim 10^{-7}$ . For smaller  $|x|$ , the Taylor series  $\sin x \approx x - \frac{1}{6}x^3$  shows that the leading-order term in  $\sin x$  is  $\gtrsim 10^{15}$  times larger than the next-order term,

which means that only one or two significant digits of the next-order term are present in the floating-point representation of  $\sin x$ . As a consequence, in this regime the function  $x - \sin x$  can only be computed to 1 or 2 significant figures by direct evaluation of the expression.

Of course, one would never evaluate the expression directly in this regime. After noticing that the calculation was suffering from this catastrophic loss of precision, one would test for the magnitude  $|x|$ , and if it smaller than some threshold like  $10^{-5}$  (say), one would replace the direct evaluation of  $x - \sin x$  by evaluation of  $\frac{1}{6}x^3$ , confident in the knowledge that the next-order term in the Taylor series being  $O(x^5)$ , one is committing at worse an error of relative magnitude  $10^{-10}$ . Switching precision from 8 to 16 bytes would be overkill in this context, and would clearly not be worth the incurred computational cost.

If this example illustrates correctly the situation with respect to the numerical origin of the “instability” phenomenon (I could be misunderstanding), then the analogous approach would be to perform an analysis of the limiting behavior of the offending terms, and when the offending regime is detected approaching, switch to evaluating a limiting expression, rather than switch floating-point precision.

Having said this, I do understand that the amplitudes  $A_k$  are in general very complicated integrals over virtual degrees of freedom, and that analogizing their cancellation pathology to the  $x - \sin x$  example is almost certainly a coarse oversimplification. I certainly do not expect the replacement of the full expression by a limiting form to be a simple exercise in elementary analysis. What I am suggesting is that it is possible that this mathematical fix may have been judged not worth the effort in the past, because switching floating-point precision was a “good enough” workaround. Going forwards, however, this choice is a crippling one with respect to the transition to HPC. If the amplitude-computing libraries could be upgraded to address the cancellations mathematically, rather than by brute force, a very serious bottleneck could be removed.

## 4 Stochastic Work Allocation

Suppose it were not possible to either refactor the transition probability calculation or to eliminate the unpredictable high-cost evaluations. Then one would be stuck with attempting to scale the existing computation in some way. The chief obstacle is how to allocate work to nodes in such a way as to minimize the risk of having a lot of nodes with low-cost work waiting on a few nodes doing high-cost work.

Paul Hovland pointed out that to some extent, the problem may be viewed as one of statistical modeling. That is, if each core has  $E$  integrand evaluations, and the distribution of high/low work evaluations is the same on each core, then as  $E$  becomes very large the amount of work per core becomes more predictable. The existing algorithm requires synchronization of Vegas histograms between cores between evaluations, so it may be difficult to exploit this increased predictability unless some kind of asynchronization of the Vegas algorithm can be implemented.

In any event, it would be very helpful to have statistics on work costs – both overall histograms and cost versus phase-space parameters data. One could in principle imagine designing and training a Deep Learning network to predict high-cost regions of the phase space, so as to increase the predictability of the unit workloads. Being able to say in advance “this evaluation has a 98% chance of being cheap/expensive” (or some similar statement) would certainly simplify the load-balancing task.